



# Efficient Group Coordination in Multicast Trees

H.-P. DOMMEL

hpdommel@cse.scu.edu

*Computer Engineering Department, Santa Clara University, Santa Clara, CA 95053, USA*

J. J. GARCIA-LUNA-ACEVES

jj@cse.ucsc.edu

*Computer Engineering Department, School of Engineering, University of California, Santa Cruz, CA 95064, USA*

**Abstract.** The majority of today's Internet applications relies on point-to-point communication. In recent years, however, multipoint communication support has become the foundation for such applications as multiparty video conferencing, distributed interactive simulations, and collaborative systems. We describe a novel protocol to coordinate multipoint groupwork within the IP-multicast framework. The protocol supports Internet-wide coordination for large and highly-interactive groupwork, relying on the dissemination of coordination directives among group members across a shared end-to-end multicast tree. We also describe how addressing extensions to IP multicast can be used for our multisite coordination mechanism.

**Keywords:** group coordination, tree-based multicast, process coordination, multiparty collaboration

## 1. Introduction

Internet-based computing is gradually migrating from the standard unicast transmission model to multicasting. In the IP multicasting model [2], a source needs to send a packet only once to the network interface, and multicast routers replicate the packet on its transmission path to multiple receivers. The Internet Group Management Protocol (IGMP) allows a host to join a multicast group by informing its local router to forward multicast traffic for this group to the leaf subnetwork where the host resides. Protocols such as DVMRP, MOSPF, PIM-SM/DM, MBGP, and MSDP [2] perform the construction of intradomain and interdomain multicast delivery trees, and enable packet forwarding among routers. However, the standard IP multicast model provides no guarantees for reliable or order-preserving delivery of packets, and every message is delivered to all members of a multicast group on a best-effort basis. A variety of protocols for multicast routing and reliable multicasting have been proposed in recent years as solutions to the numerous inherent architectural and deployment issues [4]. In this article we investigate supporting coordination among users and processes taking advantage of IP multicasting. We view group coordination among nodes in the IP multicast service model as complementary to group membership and information dissemination within the group, which are handled by the Internet Group Membership Protocol (IGMP) and multicast routing protocols (e.g., PIM, CBT, and the like), respectively. As the following sections describe, any distributed applications exerting multisite control and building on coordination of events and activities may benefit from tapping into group coordination services.

The problem of coordination of processes and their resources in network-centric computing has received considerable attention in recent years. In our context, group coordination services entail activity coordination among interacting hosts, including flow synchronization from different sources, ordered delivery of distributed event information, and the concurrent use of and access to shared resources, also referred to as floor control [5].

The origins of group coordination can be traced back to protection and access control mechanisms in operating systems [11], mutual exclusion in distributed systems [15], and concurrency control in database systems [3]. None of these mechanisms caters to continuous multimedia content or host interactivity. Studies in the context of computer supported cooperative work [6, 9] have shown that social aspects such as lack of eye contact, as well as technical factors, for example limited bandwidth and network delays, contribute to coordination problems in network-mediated collaborative work. These problems can materialize in greedy user behavior, contention for resource usage, and a lack of group focus in joint task completion. Many proposed mechanisms and systems for online collaboration [1, 8, 17–19] are proprietary, sparsely documented, and limited both in scope and scale to local area networks and few users.

We address the problem of how to allow group-centric collaborative applications to take advantage of the underlying multicast routing service and node geometry to coordinate interactions among the group members efficiently, without restrictions in scope and scale. To this end, we describe a novel tree-based group coordination mechanism called the Aggregated Coordination Protocol (ACP). ACP operates on a shared multicast tree as a shadow of the underlying multicast routing tree to store and forward coordination primitives among hosts. Multiple parties in different multicast on-tree groups are supported by ACP to coordinate their distributed activities via message passing. ACP can be used for floor control by establishing ephemeral resource usage rights through coordination primitives, rather than operating system locks, and its semantics extends to continuous media flows as well as discrete data. Raymond [15] described an ancestor protocol of ACP, where a mutual exclusion algorithm operates on a static logical propagation tree to determine the node gaining exclusive access to a critical section.

The rest of this paper is organized as follows: Section 2 presents the system model and assumptions. Section 3 describes ACP and addresses its correctness, robustness, and performance. Section 4 summarizes the benefits of tree-based group coordination.

## 2. Model and definitions

We define a coordination session  $C_S = (P, H, L)$  in a computer network to consist of processes,  $P$  coordinating a set of interactive hosts,  $H$  over links  $L \subset H \times H$ . The entities involved in a coordination process are users (agents) and resources at the end-hosts, and the ACP packets coordinating them flowing across links. Communication is by message passing only, and we assume that messages eventually arrive correctly. We refer to ACP packets as control primitives (CPs) and each host in a

session is client and server for CPs to other hosts. CPs synchronize the hosts' joint tasks, implement causal or total ordering in distributed events, and mitigate access to shared, but exclusive resources. Coordination management must be aligned with membership operations, such as joining, leaving, or splitting groups.

We will illustrate group coordination with floor control, which entails mechanisms and policies to mitigate mutually exclusive access on a set of shared resources as the coordination objective. Resources can be located at a specific end host (camera), in replicated form at every host of a multicast group handling the same resource (telepointer), or in the network (voice channel). We distinguish between generic CPs ("grant," "release," "open") and resource or media-specific CPs ("rotate left," "zoom in"). The temporary privilege to work with a multimedia resource is also called *floor*. CPs are exchanged among hosts to resolve contention for a specific resource floor, and to preserve a global activity order. CPs transfer a sender id, the receiver ids, the time of creation, a control directive, the allowed duration, and an optional priority value. Users may assume social roles (moderator, panelist, student, etc.), or control roles, which include the *floor controller* (FC) of a resource  $R$  as the host managing access and interactions for  $R$ , and the *floor holder* (FH) as the host temporarily permitted to use the resource.

Control over shared resources can be centralized, distributed, or a hybrid of both. It can be performed successively by individual session members, partitioned, where various session hosts contribute different control functions, or democratic, where consensus is achieved by negotiation, yielding a new consistent control state. A host holds a floor in each turn for a time interval  $T$ , which can be preset or timed out. In our model, we assume that the FC role is associated with a specific host, but it may infrequently rove among hosts. The FH changes at every turn. We assume that each host in a session runs the same coordination protocol, serving requests from other hosts and transmitting requests for resources placed by users.

### 3. Aggregated coordination protocol

#### 3.1. Description

The Aggregated Coordination Protocol (ACP) operates on a control tree, consisting of three main types of nodes: *holder nodes* host the FH, operating on a resource and being transmission sources; *control nodes* host the FC for a specific resource, and are addressed by other nodes asking for a floor; and *target nodes* receive updates of the operations by a FH. Nodes on the path from a holder to its targets are referred to as hop nodes. Leaf nodes terminate the downward forwarding of control information in the tree. ACP packets are assumed to be transferred reliably, which is implicitly guaranteed by the underlying reliable multicast protocol. If ACP performs independently, it needs to supply its own reliability mechanisms.

Each CP contains a version field  $V$ ; a flag field for protection bits; a *TTL* field to scope the CP; a sequence number field  $SN$  and timestamp  $TS$  to order CPs and uniquely identify them in the shared event space; an identifier  $CPid$  to uniquely tag the CP in the session; a checksum  $CS$ ; a command field  $Cmd$  for specific directives;

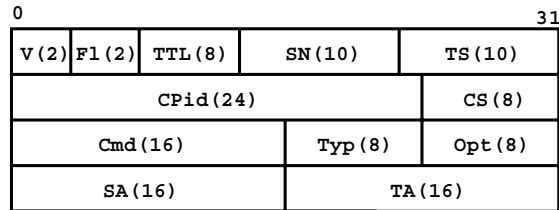


Figure 1. Packet header fields for coordination primitives (CPs).

a type field *Typ* to classify the operation; an options field *Opt* for the command; and the source address *SA* and target address *TA*, which can be a multicast group address. The 24 bits for CPids are sufficient to characterize fine-grained resource coordination in large sessions and the 16-bit prefix labeling used for *SA* and *TA* is sufficient for sessions up to 64K hosts. Figure 1 depicts this structure of a CP packet.

CPs are disseminated across a single shared tree connecting all hosts in a session. The tree corresponds to a single shared acknowledgment tree for concurrent multicasting, allowing multiple sources instead of building separate dissemination trees per source and multicast group. We outline tree maintenance for the case that group coordination is deployed in a network in conjunction with an existing underlying reliable multicast tree. The coordination tree can be a working copy of the reliable multicast tree prepared by a protocol such as Lorax [13]. Various mechanisms for initiation, joining, and advertising of collaborative sessions exist. We assume that one host, representing itself or a multicast group, initiates a session and advertises the session description, multicast address and media used in a session directory [7]. The directory serves as a rendezvous interface, which allows other hosts to join via a call-up mechanism.

The control tree is grown from the initiating node as the root, and other hosts joining the session are considered first off-tree, unicasting request-to-join messages to the inviting root node based on addressing information provided by the session directory. A TTL (time-to-live) field in the join packet restricts the session scope. A successful adoption of a child node to the control tree is confirmed with a bind message. Each newly joined host, as the root of its subtree, locally advertises invitation-to-join messages. It retrieves the current control state for resources of concern by polling its parent node. It may also be the case that separate subtrees may fission together to create a joint control tree. Each node in the tree has a maximum degree  $D$ , which must be high enough to reflect the session structure, but not exceed the capacity of a host to efficiently serve its children. Furthermore, if the tree also serves as a media mixing hierarchy [14], the permissible height must satisfy the end-to-end delay constraints.

Open sessions with dynamic membership may incur frequent joining and leaving, or accidental withdrawal of hosts from the control tree. When the root leaves the control tree, the eldest child in the subtree becomes the designated new root. Age may be determined by location, joining time, or address labels. Although the initial source and root will change during the lifetime of a session, the branching properties of the positional tree will not change, when the tree is virtually rehung with shifting

of control roles. A hop node, which lost contact with its parent for a timeout larger than possible congestion delays, must rejoin the tree as described. CPs from a node identified as lost are held at the processing host until a timeout, and are discarded if the host does not reappear. A lost and rejoined host must resend its pending CPs.

Routing of CPs in the tree is performed as follows: if a target node (FC or FH) is in the subtree of a node, the CP is routed downward the subtree branch where the target resides, otherwise it is sent upward to its parent node. The exact location of the target node is not needed since CPs are forwarded on the basis of directionality. This approach is similar in principle to Raymond's mutual exclusion solution [15], where a logical control tree is used to direct critical section requests for one resource at a time towards the node determining critical section entry. In contrast, our protocol handles multiple resources and routes are chosen based on the underlying actual multicast tree. Due to a higher degree of interactivity, where many nodes may assume FC or FH roles during session time, we introduce a more expressive addressing mechanism with self-routing semantics based on prefix matching of node labels.

The idea of using binary labels has first been used for routing of instructions in multiprocessor systems. In networking, it has been used for IP address lookup mechanisms in CIDR [16], promoting hierarchical address aggregation in small forwarding tables using prefix matching, and for router acceleration in multicast routing and reliable multicasting [12]. Prefix matching is also an elegant solution for subcasting and anycasting.<sup>1</sup>

ACP assigns unique prefix labels recursively and top-down to each node joining the control tree, such that a child node label contains the label of its parent as a prefix. For example, using a binary alphabet, the root receives label **1**, its children are numbered **10** and **11**, etc. Labels allow nodes to be addressed individually, even when belonging to more than one potentially overlapping multicast groups. Also, hosts can be placed in the tree independent of their membership in different multicast groups, in contrast to other approaches, which demand that the tree organization closely reflects group membership [10].

CPs, such as requests for the floor on a resource, are propagated across the tree using aggregation. This mechanism corresponds to solutions for the Ack implosion problem in reliable multicast, reducing the feedback of receivers to a source on lost or corrupt packets [13]. Aggregation constrains the control process to local groups, i.e., CPs with requests that can be satisfied within the neighborhood of the issuing node, will not have to be forwarded all the way to the target controller node. The controller node is hence liberated from the need to communicate with every host in the session, and deals only with requests which could not be satisfied along the feedback path. If a hop node receives the same CP from different nodes within a timeout, it aggregates them into one CP. It checks then if a response to these requests can be satisfied locally by polling its own state, and the state of neighboring nodes. Otherwise the composite CP is self-routed up, or down, in the tree toward the target node(s) as outlined before. This implies that the number of CPs required to coordinate nodes decreases as the request activity increases, because requests are not sent further, if a hop node is reached that already processed the CP.

Hosts need to maintain the following state locally: the resource ids shared locally with remote hosts, and the remote resource ids accessed locally, together with their CPids; a state table indicating which resources are locally available or held remotely, together with the id of the remote FH; and a request queue *ReqQ* which collects successive CPs from different hosts (the queue is limited by the number of nodes in the session). In addition, each node maintains a FIFO queue of pending CPids, identified by the senders' labels. A hop node receiving a CP compares the label of the target node with the head of the queue, and elects itself as receiver if its own label matches the head, or forwards it according to the routing procedure outlined previously. A control node receiving a request, responds immediately to the requester, if its local queue is empty, or it appends it to its queue. When control shifts from one node to the next, the pending request queue is transferred to the new control node, and its new address is multicast to all groups sharing the associated resource.

### 3.2. Example

Consider a scenario where three hosts from three different multicast groups, MG1–MG3, must coordinate access to a shared resource they are contending for. Figure 2 depicts a snapshot of the protocol operation in a ternary tree.

Assume that hosts **12**, **100**, and **11** all request the floor held by FC at location **101**. All request packets need to be routed along branches of the tree to **101**. The prefix property of the labels allows self-routing of these packets. For example, assume that all hosts are informed that host **101** is FC. Host **12** compares its label with the target label. Its prefix matches (**1**), but the second identifier indicates that the FC is on subtree 0. The request packet is hence sent upward to host **1**, which compares its label with the target, and, detecting that **101** is one of its children, sends the packet to host **10**, whose label matches the prefix of **101**. This node performs the same comparison and the packet ultimately arrives at FC, which finally grants the floor to node **12**.

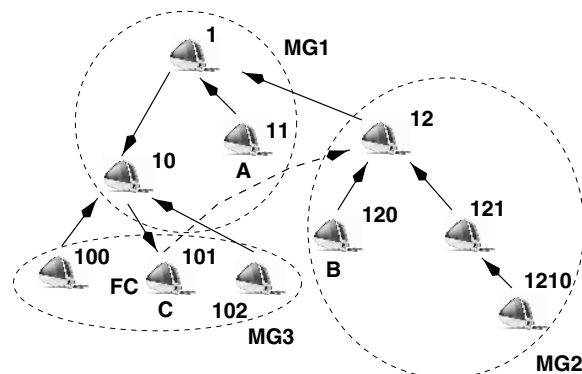


Figure 2. Snapshot of ACP operation among multicast groups MG1–MG3.

### 3.3. Correctness

We outline correctness and resilience properties of ACP. Assuming that CPs are sent reliably, a node can receive the privilege associated with a CP, only if it sends a request to the node controlling the associated resource. A message enacting the CP (e.g., grant-floor) can be sent only by a controlling node. The addressing labels in the tree are unique, and a CP is issued only for a specific requesting node. Provided that *a priori*, there is a single privileged FH per resource. We need to show that ACP will continue to forward and interpret CPs, such that the single FH property per resource is maintained.

If several nodes send a CP and another node holds the privilege associated with it for an infinite time, a deadlock exists. Reasons may be that the wrong node, or no node, holds, the privilege, it is unreachable, or the propagation of a request fails prematurely. Assume a tree with  $N$  hosts. In the worst case, each CP has to traverse  $2N$  hops for a node to get its CP request satisfied, where  $2N$  is the length of the longest path in the tree. Let us now assume that ACP operates correctly for  $N + 1$  nodes in a tree of height  $H = 2$ , with an additional root node. Using induction over  $H$ , we assume that the liveness argument holds for any level  $L$  with  $2 \leq L < H$ . A tree of height  $H + 1$  has one additional level, from where CPs can be sent and must be replied to. Assume that a node in the additional level of the tree (the root, or a leaf level of a subtree) sends a CP. The prefix matching semantics of the protocol ensures that messages are also correctly self-routed to the target address at this tree level, and will eventually reach the target node. The acyclic tree geometry prevents that a CP request can be outrun by a moving CP response (e.g., a grant message). Consequently, a host with a pending CP request cannot starve indefinitely, and ultimately gets its CP satisfied. If we consider buffering at nodes, the finite FIFO nature of request queues at each node guarantees that every CP will eventually be served. Thus, ACP operates live and correct at any tree height.

### 3.4. Fairness and resilience

Fairness refers to the average frequency and duration, by which nodes acquire a privilege for a given period. This period is, for example, the session lifetime, or the lifetime of a shared resource in a session. Network latency, geographic distance, and location of nodes, or varying host capabilities, can all be factors in causing uneven dissemination patterns for CPs, and consequently unfair allocation of resource access privileges. Leaf nodes take more time to propagate their requests across the root to a node on the other side of a tree, than nodes just below the root. Shared trees also do not provide shortest paths between a source and its receiver set [2], which may cause increased latency in CP transfer. It is hence important to establish service policies, which counteract these factors. One simple solution, a “least-recently-served” policy, can be enacted by letting each node maintain a local record of the most recent CPs and their originating nodes. Those nodes are serviced first, which do not appear on the list, or appear least in duration or frequency of service.

Previously, we assumed that transfer of CPs and accounting of control information among nodes is failure free. Even if we assume reliable multicasting to ensure that CPs are eventually transferred, the control apparatus may need additional recovery mechanisms to ensure consistency. This applies to regular node failure, control node failure, link failure, or control token loss or duplication. Such exceptions can be avoided by disseminating status information among active nodes, or by systematic detection of loss and recovery. Regular node or controller failures are typically detected via timeout, and recovered with an election protocol, with neighbor nodes providing state updates. Continuation of a split session is possible if a quorum of the members in each partition agrees to continue. One method to deal with the case that a CP is lost completely or reaches only a subset of nodes is to multicast a `CP_probe` message from a control node to the group.

### 3.5. Performance

Attaching positional labels to nodes in a  $D$ -ary tree implies an additional storage cost of  $\log_2 D$  bits per level, in a positional tree of  $N$  receivers and height  $\log_D N$ , i.e.,  $\lg N$  bits are needed. With 16-bit labels for designating sources and targets in message headers, up to  $2^{16}$  hosts can be accommodated. Prefix comparison is cheaper for nodes close to the root, due to shorter labels. Serving a CP costs  $C_{CP} = c_{req} + c_{resp} + c_{upd}$ , comprising the cost to send a request to a control node, receive a response, and multicast an update of the new state. The response time  $t_r$  to receive a response is bound by the maximum time for the probe to traverse the longest link,  $t_{max}$ , plus the time  $t_{ack}$  for the receiver nodes to send a positive or negative acknowledgment,  $t_r = 2t_{max} + t_{ack}$ . If the CP is diagnosed as lost, the controller node for the respective floor can regenerate the token and send an update to the session.

For a simple comparison of the coordination overhead incurred with unicast, multicast, and aggregated multicast, measured by the number of CPs to be exchanged, we assume full load (each node sends a CP) and a normalized processing time for request, response and update packets. The average path length between nodes is assumed to be the same for all models.  $\lambda$  represents the individual processing, packetization and transmission delay for each type of packet. In unicast, the coordination delay incurs  $(N - 1)$  requests, replies from control nodes, and updates, where  $N$  is the current session size, i.e.,  $CD_{uc} = 3(N - 1)\lambda$ . In multicast,  $(N - 1)$  nodes send requests, and the control node multicasts one reply and one update back to the session, i.e.,  $CD_{mc} = (N + 1)\lambda$ . In aggregated multicast, CPs are handled within multicast groups, and only the root of a group forwards a composite request to its parent, or responds to group-local requests, if it holds the information locally. With  $K$  groups we have on the average  $G = \lceil N/K \rceil$  members per group, and per group there are  $G$  requests inside a group,  $K$  aggregated requests sent to a control node from all groups, and one multicast response and update, i.e.,  $CD_{amc} = ((G - 1) + (K - 1) + 2)\lambda = (G + K)\lambda$ . Figure 3 shows the average cost to coordinate hosts in sessions up to size  $N = 1000$ , clustered into  $K = N/10$  groups, and with normalized transmission delay  $\lambda$ . The benefits of aggregation in conjunction with multicast are apparent.

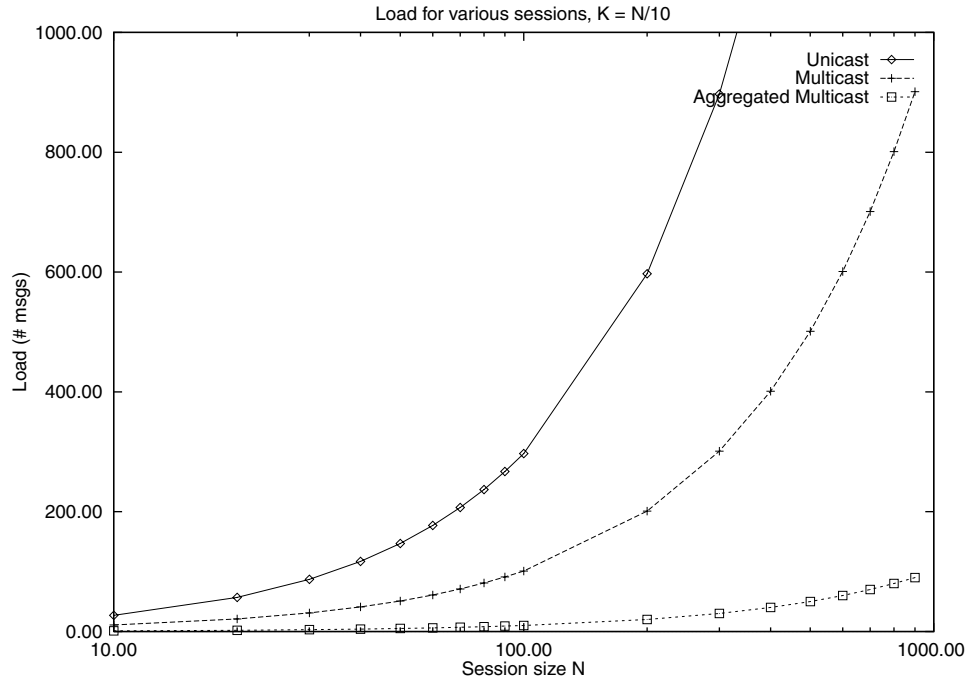


Figure 3. Coordination cost for various communication models.

#### 4. Conclusions

The IP multicast model lacks refined support for intergroup coordination. In the wake of a new generation of interactive and collaborative applications, we investigated the possibility to support group-coordinative systems with network layer functionality. To that end, we have outlined the main issues in establishing a coordination mechanism based on the IP multicast service model. The discussed protocol, ACP, operates on a logical control tree and scales to large groups and wide scope. Prefix matching previously used in CIDR addressing and multicast routing has been put to use for our multisite coordination mechanism to effectively channel the exchange of coordination messages. The proposed approach eliminates the need to build a separate control structure for tracking, routing, withholding, or forwarding control directives. A side benefit of prefix matching is the trivial provision of subcasting to and within multicast groups. We have described ACP as a mechanism, strongly coupled with the operations in an underlying routing, and end-to-end delivery tree. However, ACP may implement its own tree management at layer 4+ in an overlay network aside from multicast routing, and thus can operate independently of any layer 3 protocol. While IP multicast is still plagued by deployment and standardization problems, our protocol can be viewed as a first solution towards coordination middleware components for emerging collaborative service infrastructures in the next-generation Internet.

## Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Grant F19628-96-C-0038.

## Note

1. Subcasting delivers or retrieves data between a source and select members of a multicast group, and anycasting transfers data to any one member of a group, for example the nearest proxy in a group of servers.

## References

1. L. Aguilar, J. J. Garcia-Luna-Aceves, D. Moran, E. J. Craighill, and R. Brungardt. Architecture for a multimedia teleconferencing system. In: *Proc. ACM SIGCOMM*, pp. 126–136, 1986.
2. K. Almeroth. The evolution of multicast: from the Mbone to interdomain multicast to Internet2 deployment. *IEEE Network*, 14(1):10–20, 2000.
3. N. S. Barghouti and G. E. Kaiser. Concurrency control in advanced database applications. *ACM Computing Surveys*, 23(3):269–317, 1991.
4. C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network*, 14(1):78–88, 2000.
5. H.-P. Dommel and J. J. Garcia-Luna-Aceves. Floor control for multimedia conferencing and collaboration. *Multimedia Systems J. (ACM/Springer)*, 5(1):23–38, 1997.
6. C. A. Ellis and S. J. Gibbs. Concurrency control in groupware systems. In: *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Portland, OR, pp. 399–407, 1989.
7. M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327, 1998.
8. International Telecommunication Union. Recommendation T.120 on Data protocols for multimedia conferencing. <http://www.itu.int>. 1996.
9. E. A. Isaacs, T. Morris, T. K. Rodriguez, and J. C. Tang. A Comparison of face-to-face and distributed presentations. In: *Proc. ACM CHI*, Denver, CO, 1995.
10. X. Jia. A total ordering multicast protocol using propagation trees. *IEEE Trans. Parallel and Distrib. Sys.* 6(6):617–627, 1995.
11. B. Lampson. Protection. *Proc. 5th Princeton Conf. on Information Sciences and Systems*, 8(1):18–24, 1974.
12. B. N. Levine and J. J. Garcia-Luna-Aceves. Improving internet multicast with routing labels. In: *Proc. IEEE Int. Conf. on Network Protocols*, Atlanta, GA, pp. 241–250, 1997.
13. B. N. Levine, D. Lavo, and J. J. Garcia-Luna-Aceves. The case for reliable concurrent multicasting using shared ack trees. In: *Proc. ACM Multimedia*, Boston, MA, pp. 365–376, 1996.
14. P. V. Rangan, H. M. Vin, and S. Ramanathan. Communication architectures and algorithms for media mixing in multimedia conferences. *IEEE/ACM Trans. on Networking*, 1(1):20–30, 1993.
15. K. Raymond. A tree-based algorithm for distributed mutual exclusion. *ACM Trans. on Comp. Sys.* 7(1):61–77, 1989.
16. M. A. Ruiz-Sanchez, E. W. Biersack, and W. Dabbous. Survey and taxonomy of IP address lookup algorithms. *IEEE Network*, 15(2):8–23, 2001.
17. H. M. Vin, P. V. Rangan, and S. Ramanathan. Hierarchical conferencing architectures for inter-group multimedia collaboration. In: *ACM SIGOIS Bull., Proc. Org. Comp. Sys.*, Atlanta, GA, pp. 43–54, 1991.
18. R. Yavatkar and K. Lakshman. Communication support for distributed collaborative applications. *Multimedia Systems J.* 2(2):74–88, 1994.
19. C. Ziegler, G. Weiss, and E. Friedman. Implementation mechanisms for packet switched voice conferencing. *IEEE J. on Sel. Areas in Comm.* 7(5):698–706, 1989.